Attorney Docket No: 08831.0067

UNITED STATES PATENT APPLICATION

FOR

SYSTEMS AND METHODS FOR IMPROVED GAMUT MAPPING FROM ONE IMAGE DATA SET TO ANOTHER

BY

Michael Francis HIGGINS

AND

Candice Hellen BROWN ELLIOTT

SYSTEMS AND METHODS FOR IMPROVED GAMUT MAPPING FROM ONE IMAGE DATA SET TO ANOTHER

BACKGROUND

[01] In commonly owned United States Patent Applications: (1) United States Patent Application Serial No. 09/916,232 ("the '232 application"), entitled "ARRANGEMENT OF COLOR PIXELS FOR FULL COLOR IMAGING DEVICES WITH SIMPLIFIED ADDRESSING," filed July 25, 2001; (2) United States Patent Application Serial No. 10/278,353 ("the '353 application"), entitled "IMPROVEMENTS TO COLOR FLAT PANEL DISPLAY SUB-PIXEL ARRANGEMENTS AND LAYOUTS FOR SUB-PIXEL RENDERING WITH INCREASED MODULATION TRANSFER FUNCTION RESPONSE," filed October 22, 2002; (3) United States Patent Application Serial No. 10/278,352 ("the '352 application"), entitled "IMPROVEMENTS TO COLOR FLAT PANEL DISPLAY SUB-PIXEL ARRANGEMENTS AND LAYOUTS FOR SUB-PIXEL RENDERING WITH SPLIT BLUE SUB-PIXELS," filed October 22, 2002; (4) United States Patent Application Serial No. 10/243,094 ("the '094 application), entitled "IMPROVED FOUR COLOR ARRANGEMENTS AND EMITTERS FOR SUB-PIXEL RENDERING," filed September 13, 2002; (5) United States Patent Application Serial No. 10/278,328 ("the '328 application"), entitled "IMPROVEMENTS TO COLOR FLAT PANEL DISPLAY SUB-PIXEL ARRANGEMENTS AND LAYOUTS WITH REDUCED BLUE LUMINANCE WELL VISIBILITY," filed October 22, 2002; (6) United States Patent Application Serial No. 10/278,393 ("the '393 application"), entitled "COLOR DISPLAY HAVING HORIZONTAL SUB-PIXEL ARRANGEMENTS AND LAYOUTS," filed October 22, 2002; (7) United States Patent Application Serial No. 01/347,001 ("the '001 application") entitled "IMPROVED SUB-PIXEL ARRANGEMENTS FOR STRIPED DISPLAYS AND METHODS AND SYSTEMS FOR SUB-PIXEL RENDERING SAME," filed January 16, 2003,

each of which is herein incorporated by reference in its entirety, novel sub-pixel arrangements are disclosed for improving the cost/performance curves for image display devices.

[02] For certain subpixel repeating groups having an even number of subpixels in a horizontal direction, the following systems and techniques to affect improvements, e.g. proper dot inversion schemes and other improvements, are disclosed and are herein incorporated by reference in their entirety: (1) United States Patent Application Serial Number 10/456,839 entitled "IMAGE DEGRADATION CORRECTION IN NOVEL LIQUID CRYSTAL DISPLAYS"; (2) United States Patent Application Serial No. 10/455,925 entitled "DISPLAY PANEL HAVING CROSSOVER CONNECTIONS EFFECTING DOT INVERSION"; (3) United States Patent Application Serial No. 10/455,931 entitled "SYSTEM AND METHOD OF PERFORMING DOT INVERSION WITH STANDARD DRIVERS AND BACKPLANE ON NOVEL DISPLAY PANEL LAYOUTS"; (4) United States Patent Application Serial No. 10/455,927 entitled "SYSTEM AND METHOD FOR COMPENSATING FOR VISUAL EFFECTS UPON PANELS HAVING FIXED PATTERN NOISE WITH REDUCED QUANTIZATION ERROR"; (5) United States Patent Application Serial No. 10/456,806 entitled "DOT INVERSION ON NOVEL DISPLAY PANEL LAYOUTS WITH EXTRA DRIVERS"; (6) United States Patent Application Serial No. 10/456,838 entitled "LIQUID CRYSTAL DISPLAY BACKPLANE LAYOUTS AND ADDRESSING FOR NON-STANDARD SUBPIXEL ARRANGEMENTS"; (7) United States Patent Application Serial No. 10/696,236 entitled "IMAGE DEGRADATION CORRECTION IN NOVEL LIQUID CRYSTAL DISPLAYS WITH SPLIT BLUE SUBPIXELS", filed October 28, 2003; and (8) United States Patent Application Serial No. 10/807,604 entitled "IMPROVED TRANSISTOR BACKPLANES

FOR LIQUID CRYSTAL DISPLAYS COMPRISING DIFFERENT SIZED SUBPIXELS", filed March 23, 2004.

[03] These improvements are particularly pronounced when coupled with sub-pixel rendering (SPR) systems and methods further disclosed in those applications and in commonly owned United States Patent Applications: (1) United States Patent Application Serial No. 10/051,612 ("the '612 application"), entitled "CONVERSION OF RGB PIXEL FORMAT DATA TO PENTILE MATRIX SUB-PIXEL DATA FORMAT," filed January 16, 2002; (2) United States Patent Application Serial No. 10/150,355 ("the '355 application"), entitled "METHODS AND SYSTEMS FOR SUB-PIXEL RENDERING WITH GAMMA ADJUSTMENT," filed May 17, 2002; (3) United States Patent Application Serial No. 10/215,843 ("the '843 application"), entitled "METHODS AND SYSTEMS FOR SUB-PIXEL RENDERING WITH ADAPTIVE FILTERING," filed August 8, 2002; (4) United States Patent Application Serial No. 10/379,767 entitled "SYSTEMS AND METHODS FOR TEMPORAL SUB-PIXEL RENDERING OF IMAGE DATA" filed March 4, 2003; (5) United States Patent Application Serial No. 10/379,765 entitled "SYSTEMS AND METHODS FOR MOTION ADAPTIVE FILTERING," filed March 4, 2003; (6) United States Patent Application Serial No. 10/379,766 entitled "SUB-PIXEL RENDERING SYSTEM AND METHOD FOR IMPROVED DISPLAY VIEWING ANGLES" filed March 4, 2003; (7) United States Patent Application Serial No. 10/409,413 entitled "IMAGE DATA SET WITH EMBEDDED PRE-SUBPIXEL RENDERED IMAGE" filed April 7, 2003, which are hereby incorporated herein by reference in their entirety.

[04] Improvements in gamut conversion and mapping are disclosed in commonly owned and co-pending United States Patent Applications: (1) United States Patent Application

Serial No. 10/691,200 entitled "HUE ANGLE CALCULATION SYSTEM AND METHODS", filed October 21, 2003; (2) United States Patent Application Serial No. 10/691,377 entitled "METHOD AND APPARATUS FOR CONVERTING FROM SOURCE COLOR SPACE TO RGBW TARGET COLOR SPACE", filed October 21, 2003; (3) United States Patent Application Serial No. 10/691,396 entitled "METHOD AND APPARATUS FOR CONVERTING FROM A SOURCE COLOR SPACE TO A TARGET COLOR SPACE", filed October 21, 2003; and (4) United States Patent Application Serial No. 10/690,716 entitled "GAMUT CONVERSION SYSTEM AND METHODS" filed October 21, 2003 which are all hereby incorporated herein by reference in their entirety.

[05] Additional advantages have been described in (1) United States Patent Application Serial No. 10/696,235 entitled "DISPLAY SYSTEM HAVING IMPROVED MULTIPLE MODES FOR DISPLAYING IMAGE DATA FROM MULTIPLE INPUT SOURCE FORMATS", filed October 28, 2003 and (2) United States Patent Application Serial No. 10/696,026 entitled "SYSTEM AND METHOD FOR PERFORMING IMAGE RECONSTRUCTION AND SUBPIXEL RENDERING TO EFFECT SCALING FOR MULTI-MODE DISPLAY" filed October 28, 2003.

[06] Additionally, these co-owned and co-pending applications are herein incorporated by reference in their entirety: (1) United States Patent Application Serial No. [ATTORNEY DOCKET NUMBER 08831.0064] entitled "SYSTEM AND METHOD FOR IMPROVING SUB-PIXEL RENDERING OF IMAGE DATA IN NON-STRIPED DISPLAY SYSTEMS"; (2) United States Patent Application Serial No. [ATTORNEY DOCKET NUMBER 08831.0065] entitled "SYSTEMS AND METHODS FOR SELECTING A WHITE POINT FOR IMAGE DISPLAYS"; (3) United States Patent Application Serial No. [ATTORNEY DOCKET NUMBER 08831.0066]

entitled "NOVEL SUBPIXEL LAYOUTS AND ARRANGEMENTS FOR HIGH BRIGHTNESS DISPLAYS"; (4) United States Patent Application Serial No. [ATTORNEY DOCKET NUMBER 08831.0068] entitled "IMPROVED SUBPIXEL RENDERING FILTERS FOR HIGH BRIGHTNESS SUBPIXEL LAYOUTS"; which are all hereby incorporated by reference. All patent applications mentioned in this specification are hereby incorporated by reference in their entirety.

BRIEF DESCRIPTION OF THE DRAWINGS

[07] The accompanying drawings, which are incorporated in, and constitute a part of this specification illustrate exemplary implementations and embodiments of the invention and, together with the description, serve to explain principles of the invention.

[08] **FIG. 1** is an overview of one embodiment of an architecure of an RGB to RGBW converter.

[09] **FIG. 2** is an embodiment of a simplified RGB to luminosity converter.

[010] **FIG. 3** is an embodiment of an simplified RGB to chrominance converter.

[011] **FIG. 4** is an embodiment of a hue angle calculator.

[012] **FIG. 5** is a portion of a hue angle calculator.

[013] **FIG. 6** is one stage of a division unit embodiment.

[014] **FIG. 7** is one embodiment of a five division units connected to perform a 5-bit divide

[015] **FIG. 8** is RG case of a 3x3 matrix multiplier simplified embodiment.

[016] **FIG. 9** is the GB case of a 3x3 matrix multiplier simplifed embodiment.

[017] **FIG. 10** is the BR case of a 3x3 matrix multiplier simplified embodiment.

[018] **FIG. 11** is one embodiment of a gamut claming circuit.

[019] **FIG. 12** is one embodiment of a W selector.

[020] **FIG. 13** is one embodiment of a diagram showing reduced bandwith by delaying the W selection.

[021] **FIG. 14** is one embodiment of a diagram showing RGBW conversion and SPR in hardware.

[022] **FIG. 15** is one embodiment of a diagram showing a software implementaion of RGBW and SPR with simplified display hardware.

[023] **FIG. 16** is an alternate embodiment of a software implimentaion of RGBW and SPR

DETAILED DESCRIPTION

[024] Reference will now be made in detail to implementations and embodiments, examples of which are illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts.

Architecture Overview

[025] Several systems and methods for gamut mapping from one color space to another were disclosed in the above incorporated '200, '377, '396 and the '716 applications. The present application improves upon those systems and methods by disclosing additional savings, efficiencies and cost reduction in both the hardware and the software implementations of those systems.

[026] One potential simplifying assumption that may lead to efficiencies is assuming that the target color space is RGBW. Given that assumption, there are many optimizations possible in the "gamut pipeline". For example, for a RGB to RGBW gamut mapping system, gamut expansion may not be very important or applicable; but gamut clamping might be desired after gamut conversion.

Additionally, for multiprimary systems (e.g. RGBC or the like) that employ the 3x4 matrix multiply normally required for 4-primary multi-primary can be replaced by a 3x3 matrix multiplier and a MUX. Also, those 3x3 matrices, according to disclosed methods here and in application mentioned above, may have many elements that are zeros, ones, and/or powers of two. Such conditions may allow special purpose hardware to do the matrix multiplies with a reduced number of gates.

[027] Figure 1 shows one possible gamut mapping system 100 from a RGB color space to a RGBW color space. RGB data input 102 (possibly 8 bits per color) is input into a chroma/luma converter 104. The output of block 104 could be one of a number of chrominance/luminance coordinates (e.g. Y, By,Ry or the like) and input into a hue angle calculator 106 which then could be passed to a hue angle triangle look-up table 108 (as previously described in the above incorporated applications). The hue angle triangle could then be converted into a multi-primary matrix look-up table 110 which loads matrix coefficients into 3x3 matrix multiply 112. It will be appreciated that the specification of the number of bits per line – in either the specification or the accompanying figures – are offered only to elucidate possible embodiment; but that the scope of the present invention is not limited to any particular data set or specified bandwidth on any data path.

Chroma Luma Converter

[028]The "NTSC" formula for calculating luminosity is:

$Y=0.2126*R+0.7152*G+0.0511*B$. However, for the purposes of calculating hue angle, the formula $Y=(2*R+5*G+B)/8$ may suffice as close enough and it can be advantegously calculated using only binary shifts and adds. This is equivalent to calculating $Y=0.25*R+0.625*G+0.125*B$.

[029] Figure 2 shows one embodiment of a high level block diagram 200 that implements Y calculation as above. RGB data is input and the R is left shifted by 1 bit (i.e. multiply by 2) at 202, G

data is left shifted 2 bits and added to itself (i.e. multiply by 5) at 204 and B data is added to the red which is then summed with the green total. The resulting sum is then right shifted by 3 bits at 206 to supply the final Y data as given above. It should be appreciated that the operation and their orders may be changed, as long as the arithmetic result is the same.

[030] Figure 3 shows one embodiment of a chroma calculating block 300. The chroma component is a vector, two signed numbers, calculated from the formulae x=B-Y and y=R-Y. However, it may be desirable to use the absolute value of these numbers, so it is possible to calculate their absolute values directly and keep the signs separate. Comparators 302 and 304 are used to determine if B>Y and if R>Y, respectively. The results of these are saved as the signs of x and y for the hue angle calculator and also used to selectively swap the values before subtracting them. Subtraction may be accomplished as a twos compliment NEG operation 306 followed by an addition 308. As the values input are signed numbers, the NEG operation may result in an additional bit. However, this bit may be ignored in the addition since the sign is known to be one and the result is known to be a positive number. Alternatively, this functionality could be accomplished in a number of different ways, including to perform all possible subtractions for both values and select the positive ones at the end.

Hue Angle Calculator

[031] It may be possible to combine the chroma/luma converter with the hue angle calculator and achieve certain optimizations. Figure 4 depicts one embodiment of such a combined hue angle calculator 400.

Absolute Value of Chroma

[032] If the chroma/luma converter is combined with the hue angle calculator (as in blocks 402 and 404), the absolute values of the chroma are already available, including the signs as they would have been before taking the absolute values. Taking the absolute value helps to limit calculations to one quadrant of the possible color vector angles. It will be appreciated that the "Y" in blocks 402 and 404 refer to the luminance value; while "y" output from block 404 onward refers to a chrominance value.

Select Octant, y>x

[033] The test as to whether the chroma y value is greater than the chroma x value may determine whether the hue angle is in the first or second octant of the vector angle or, alternatively, whether the angle is greater than 45 degrees. By swapping the x and y components of chroma (as possibly performed by block 406 in Figures 4 and 5), it is possible to limit the calculations to the first octant of all possible color vector angles. Of course, the result of the test may be saved for correcting the final output hue angle. Division module 408 supplies input data to the arctangent look up table, as will be discussed later.

Action LUT

[034] One embodiment of Action LUT 410 may comprise a small table of bits and offsets that are added in the final step to correct for the simplification of doing all the calculations in the first octant. One possible embodiment of the Action LUT is included below. In this example, the concatenation of the y<0 x<0 and y>x results is the address of this LUT. The output is a "neg" bit and an offset. The neg bit indicates if the negative of the arc tangent result is needed. The offset is an

angle to add to the upper bits in the final step. It may be desirable to select the units of angle for the hue angle to produce only 256 "degrees" of angle around the color vector circle. This results in several convenient optimizations. One of these is that all the offsets in the Action LUT are multiples of 64 and the lower 6 bits were always zero and these did not need to be stored.

Action LUT

Address	neg	Offset
000	0	00
001	1	01
010	1	10
011	0	01
100	1	00
101	0	11
110	0	10
111	1	11

y/x Divide

[035] At block 408, the y component is divided by the x component of chroma. This can be done in many possible ways. One way might be to invert the x component into a fixed point fraction and then do a multiply with y. The inversion could be done in a LUT, however the results of the multiply may be inaccurate unless the multiply is sufficiently wide (e.g. 12 bits). It may be possible to accomplish the divide in a multiple step pipeline, using a module 600 ("DIV1") as shown in Figure 6. Each step in the division does a single shift, addition and selection. The output is the remainder

for the next step and one bit of the result. After a finite number of steps, all the bits needed from the division will be available.

[036] Figure 7 shows one possible embodiment 700 where x and y are 8bit data units and the result is a 5bit number. It should be noted that –x may be 9 bits, formed from an 8bit number that has been negated (twos compliment). When y is left shifted, it also becomes 9bits for the addition. Only the lower 8bits of the result may suffice for the Y OUT. Additionally, it might be possible to drop the least significant bit from both x and y, to allow using 8bit adders in all the DIV1 modules. The carry bit from the addition may be used to select either the input y value or the "subtracted" y value as the output. The inverse of the carry is the result bit.

[037] In Figure 7, it is noted that the x component of chroma need only be negated (twos compliment) once at the start of the pipeline. It is also notes that the DIV1 module has been optimized for the case when x>y and when the result is a fixed point fraction less than 1. When x=0, the result will be zero. When x=y, the result would be all bits on, which is slightly smaller than the correct answer. However, as storing the correct answer would require another bit in the result, this small error may suffice to be close enough for hue angle calculations to realize some hardware efficiencies.

Arc Tangent LUT

[038] The result of the division may be used as the index to an arc tangent table. One possible embodiment of the arc tangent table is shown below. As this table may be small, it may be possible to store both the positive and negative arc tangent values and use the **neg** bit from the Action LUT as the least significant bit of the address of the Arc Tangent LUT. In one embodiment in which the original values are 5 bit unsigned integers, their negatives may produce 6 bits to have room for

the sign bit. However, the sign bit is typically identical to the input **neg** bit, so it may not necessary to store it and the table may remain 5bits wide.

Arc Tangent LUT

00	00	01	31	03	29	04	28
05	27	06	26	08	24	09	23
10	22	11	21	12	20	13	19
15	17	16	16	17	15	18	14
19	13	20	12	21	11	22	10
23	09	24	08	25	07	25	07
26	06	27	05	28	04	29	03
29	03	30	02	31	01	31	01

Final Addition

[039] The result of the Arc Tangent LUT may be added to the offset selected from the Action LUT. However, this operation may be simpler than a full addition. Because the offset from the Action LUT may have a certain number of (e.g. 6) implied bits of zeros, the lower bits are not involved in the addition. To construct the final hue angle, the number of (e.g. 5) bits output by the Arc Tangent LUT are simply copied into the lower number of (e.g. 5) bits of the hue angle. The **neg** bit becomes the last (e.g. 6^{th}) bit of hue angle, and additional (e.g. two) more copies of the **neg** bit are added to the offset bits from the action table to form the upper (e.g. two) bits of hue. Thus, in this one embodiment, only a two bit addition is necessary. This is shown in the following table.

	neg	neg	neg	atan4	atan3	atan2	atan1	atan0
+	off1	off0						
=	hue7	hue6	hue5	hue4	hue3	hue2	hue1	hue0

Chromaticity Triangle LUT

[040] The hue angle may be used as the index to a table to determine which chromaticity triangle the input color lies in. One embodiment of chromaticity triangle LUT is given below. In the case of RGBW, there may be only three chromaticity triangles, so the table may result in only one of three possible values. The calculations leading up to this look-up may trade-off the need for a larger LUT without such calculations.

Chromaticity Triangle LUT

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2

Multi-Primary Matrix LUT

[041] The chromaticity triangle number may, in turn, be used to select one of the multi-primary matrices, stored in LUT 110 in Figure 1, to be used in a color-space conversion step later. These numbers may change according to the characteristics of any given, different, display model – one embodiment of which is shown below. It should be noted that the conversion matrices may involve positive and negative numbers, so the multipliers may be signed -- unless optimizations suggested herein are used. In one embodiment, the values in these matrices may be multiplied by 128 to allow room for 7 bits of value plus a sign bit. Thus, the results may be divided by 128 instead of 256.

Multi-Primary Matrices

168 0 -40 0 168 -40 0 0 128 RGW, triangle 0	128 0 0 -40 168 0 -40 0 168 GBW, triangle 1	168 -40 0 0 128 0 0 -40 168 BRW, triangle 2

RGB Color Path

Input Gamma LUT

[042] In one embodiment, incoming data to the pipeline could be "sRGB", or nonlinear RGB. In such a case, it may be desirable to linearized this data in an (optional) input gamma table (as shown as block 103 in Figure 1) before performing color conversions or sub-pixel rendering. It

should be noted that the hue angle may be calculated from the sRGB values, since the color conversion should preserve hue angle. This allows hue angle to be calculated with the nonlinear RGB values. One aspect of sRGB is that it acts somewhat as a compression scheme that allows image data to be stored in 8bits when it might normally require more. So, once the data is linearized, it may be desirable to store the resulting data in more bits to avoid any possible image defects. Thus, in the embodiment shown in Figure 1, input gamma block 103 converts the 8bit input data to 11bit linear RGB data.

[043] If the input data turns out to be YCbCr or some other TV format, most of these also have an implied nonlinear transformation applied to them and may also require an input gamma table. For these formats, it may be desirable to convert into sRGB before sending it down the pipeline.

Multi-Primary Matrix Multiply Conversion

[044] In one embodiment, it is possible to perform a 3x4 matrix multiply in order to effect RGB to RGBW color-space conversion. This might require 12 multipliers and adders. However, in RGBW, the W value may turn out to be equal to one of the other results, reducing the matrix multiply down to 3x3. In one embodiment, this may still be problematical to implement since each multiply is 11x8=12 bits with the 8bit coefficient signed as well. It should be noted that the multipliers input 11bit values but output 12bit results. This extra bit may be used to detect out-of-gamut values in the gamut clamping path described below.

[045] Advantegeously, many of the coefficients in the matrices are either zero or powers of two. Of the remaining coefficients, multiplying by 168 can be done with three shifts and adds while 40 can be done by two shifts and adds. To take advantage of these constants, special purpose

hardware can be designed for each chromaticity triangle. Fortunately, in RGBW, there are only three triangles so the hardware to do all the cases may remain simple. It is possible that all three formula will run in parallel with a MUX at the end to select the correct answer based on the chromaticity triangle number output by the hue angle path. Figures 8, 9 and 10 are embodiments that implement the calculation for the RGW, GBW and the BRW chromaticity triangles, respectively. Thus, in these embodiments, multiplying the 11 bit input numbers by the 8bit constants will result in 19bit numbers. When these are right shifted by 7, the final result will be a 12bit number. The upper bit on these values indicates that the color is out-of-gamut and is used by the gamut clamping path described later. It is also possible to generate negative results here, and these must be clamped to zero.

[046] It should also be noted that each input color is multiplied by 168 in two of the three triangle cases. This calculation could be shared between the formula, only multiplying by 168 a total of three times, further reducing the total number of gates. It should also be noted that the exact constants used may change when the chromaticity of each new RGBW display model is measured.

Gamut Clamping Path

[047] When black and white are mapped to the same colors in RGB and RGBW, the total gamut "volume" of RGBW may turn out to be smaller than RGB. Thus, there may be some colors, especially bright saturated ones, that exist in RGB but cannot be displayed in RGBW. When these colors appear, it may be desirable to manage this situation. Simply clamping the RGBW values to the maximum range may result in the hue of these colors being distorted. Instead the out-of-gamut colors could be detected and scaled in a way that preserves hue while bringing them back into range.

Detecting In-Gamut

[048] The multipliers and accumulators in the multi-primary matrix conversion section above may be designed to return values larger than their input values. This is to allow out-of-gamut (O.O.G.) values to be calculated. These values are typically not more than twice the range of the input values, so one more bit may be allowed in the output for "overflow" values. If this extra overflow bit is zero in all three of the R G and B results, then the color is in gamut and it could be gated around the rest of the gamut clamping path. Figure 11 shows one embodiment of hardware that could effect the functionality of blocks 114 and/or 116 in Figure 1. As may be seen, the upper bit (bit 11) of all three converted primaries are OR'ed (1102) together to produce the O.O.G. signal – which can then be used by multiplexors 1110 to select a bypass mode or data modified by the Inv LUT 1106.

Out-Of-Gamut Response

[049] If the overflow bit in any one of the R G and B results is on, this indicates that an out-of-gamut color has resulted and all three of the primaries may be scaled by some factor – e.g. the same factor. Scaling all three components by the same factor may tend to decrease luminosity and saturation but preserve hue. This scale factor typically is a number slightly less than one, so it may be a fixed point binary fraction.

Maximum Component

[050] One manner of handling out of gamut data is to calculate the ratio of distance to the edge of the gamut relative to the out-of-gamut distance as the gamut scaling factor to bring out-of-

gamut values back in range. In one mode of calculation, this might require calculating two square roots. In another embodiment, the ratio of the width of the color-space relative to the maximum component of the out-of-gamut color may yield the same result -- without need of costly square root calculations. This may be seen by looking at similar triangles within the gamut. The width of the color-space tends to be a power of two (e.g. 2^{11} for the case of 11 bit linear RGB values) and becomes a convenient bit shift. MAX block 1104 selects the maximum component of the out-of-gamut color.

Inverse LUT

[051] The maximum out-of-gamut component is inverted by looking it up in an inverse LUT 1106. In one embodiment, although using 12bit converted values will allow 2-times out of gamut values, in practice, it may be rare that it will be more than 25% above the maximum allowed value. This allows the Inverse LUT to have only 256 entries. The lower 8bits of the maximum out-of-gamut component may be used as an index into this table. A table of inverses may contain some errors, but the first 25% of the 1/x table is typically not where the errors occur, so this may suffice.

Clamping Multipliers

[052] In one embodiment, the Inverse LUT may have 12bit values in it, so three 12x11=11 multipliers may suffice to scale out-of-gamut values back down into range. The output of the multipliers may only be 11 bits because the inverse numbers could be expressed as fixed point binary numbers between 0.75 and 1. It is also possible that the inverse table could be a little narrower, perhaps only 8bits per inverse value, resulting in significant savings in gates by using a 12x8=11 multiplier.

[053] When the R G and B components output from the multi-primary matrix multiply are out-of-gamut, they may be multiplied by the output of the Inverse LUT. When the value is in gamut, the input values may be gated around the multipliers, thus bypassing the gamut clamping.

White Selection

[054] As mentioned above, the W value of RGBW may turn out to be equal to one of the other primaries, so selecting W may be delayed until later to avoid duplicate processing. Figure 12 shows one embodiment of hardware that selects the W value from one of the other converted primaries with a MUX. The result will be 4 primaries, RGB and W and this concludes the RGB to RGBW multi-primary conversion. It should be noted that the W value is equal to one of the other primaries up to this stage, but since the Sub-Pixel Rendering treats W different than the other primaries, the final results to the display will be a W value different than any of the other primaries.

Sub-Pixel Rendering and Output Gamma

[055] In one embodiment, the output from multi-primary conversion may be linear color components so the sub-pixel rendering module will not have to perform input gamma conversion. This also means that the input components may have more than 8bits per primary (e.g. 11bits in one embodiment). In the embodiment of Figure 1, there is shown output gamma being performed after the sub-pixel rendering to show that the data can stay in the linear domain until the last moment before being converted to send to the display. It should be appreciated that such an output gamma table may be tailored for the particular display panel.

Optional Output Gamma LUT

[056] In other embodiments, it is possible that the RGBW display may employ more than one step on more than one board. Thus, between boards, it may be desirable to transmit the data on standard interfaces with 8bit values. As mentioned above, truncating the linear components to 8bits is not preferred. One manner to compensate is to convert the data for transmission by applying the sRGB non-linear transformation to the data on the way out. Then, the second board can perform input gamma correction to linearize the data again to 11 bits.

[057] It may also be difficult to send 4-primary colors between the boards. Figure 13 depicts one embodiment. The system sends two bits of information along with three (RGB) primary colors, the W selection MUX can be moved onto the second board and the W primary will not have to be transmitted between boards. The two bits of information sent would be the chromaticity triangle number calculated on the hue angle path.

RGBW Simplified for Low Cost Implementations

[058] The complexity of doing multi-primary conversions seems to have confined RGBW to used only in high-end systems. However, there may be ways to use the multi-primary conversions for RGBW in low cost displays. The few remaining multiplies by odd constants may be done in software in some implementations, or perhaps it is suffices to convert those constants into numbers that are easier to implement in hardware.

[059] When the primaries and white point are identical to the sRGB standard, the matrices become even simpler. The sRGB primaries and white point results in numbers that can be multiplied with only 2 or 3 shifts and adds as shown above and in Figures 8, 9 and 10. The limiting factor may be the complexity of the SPR algorithms.

	Red	Green	Blue	White Point
x	0.6400	0.3000	0.1500	0.3127
y	0.3300	0.6000	0.0600	0.3290
z	0.0300	0.1000	0.7900	0.3583

[060] The above table has the CIE Chromaticity values for the sRGB standard. Using these values the CIE XYZ coordinates of the D65 white point can be calculated and the conversion matrix for converting linear RGB values into CIE XYZ tristimulus values can be derived:

$$D65 = \begin{pmatrix} 0.950456 \\ 1 \\ 1.089058 \end{pmatrix}, \qquad R2X = \begin{pmatrix} 0.412391 & 0.357584 & 0.180481 \\ 0.212639 & 0.715169 & 0.072192 \\ 0.019331 & 0.119195 & 0.950532 \end{pmatrix},$$

[061] Additionally, one possible matrix that converts RGBW values into CIE XYZ tristimulus values using the above primaries is as follows:

$$W2X = \begin{pmatrix} 0.314179 & 0.272425 & 0.137499 & 0.226353 \\ 0.161998 & 0.54485 & 0.055 & 0.238153 \\ 0.014727 & 0.090808 & 0.724161 & 0.259362 \end{pmatrix},$$

[062] The matrices that convert CIE XYZ tristimulus values into RGBW values are given below as:

$$Mrg = \begin{pmatrix} 4.236707 & -1.954206 & -0.984886 \\ -1.289617 & 2.526155 & -0.275862 \\ 0.05563 & -0.203977 & 1.056972 \\ 0.05563 & -0.203977 & 1.056972 \end{pmatrix}, \quad Mgb = \begin{pmatrix} 3.24097 & -1.537383 & -0.498611 \\ -2.285349 & 2.942975 & 0.21041 \\ -0.940103 & 0.212844 & 1.543245 \\ 3.24097 & -1.537383 & -0.498611 \end{pmatrix}, \quad Mbr = \begin{pmatrix} 4.557083 & -2.604397 & -0.667467 \\ -0.969244 & 1.875968 & 0.041555 \\ 0.376004 & -0.854165 & 1.374389 \\ -0.969244 & 1.875968 & 0.041555 \end{pmatrix},$$

[063] An input color would be converted using one of these three matrices, depending on which chromaticity triangle it lies in. These coefficients may be derived using the standard sRGB chromaticities. Using the same primaries for the input data and the display simplify these matrices.

[064] When the color primary assumptions of an input image are not known, sRGB assumptions may be used. Input RGB values would be converted to CIE XYZ by using the R2X matrix mentioned earlier, then converted to RGBW using one of the three matrices above. In practice, the R2X matrix can be combined with each of the other three matrices beforehand so that only one matrix multiply suffices for each input color. Also in a low cost implementation the matrices are converted to integers by multiplying them by some power of two:

$$MRG := Mrg \cdot R2X \cdot 64 \qquad MRG = \begin{pmatrix} 84 & 0 & -20 \\ 0 & 84 & -20 \\ 0 & 0 & 64 \\ 0 & 0 & 64 \end{pmatrix},$$

$$MGB := Mgb \cdot R2X \cdot 64 \qquad MGB = \begin{pmatrix} 64 & 0 & 0 \\ -20 & 84 & 0 \\ -20 & 0 & 84 \\ 64 & 0 & 0 \end{pmatrix},$$

$$MBR := Mbr \cdot R2X \cdot 64 \qquad MBR = \begin{pmatrix} 84 & -20 & 0 \\ 0 & 64 & 0 \\ 0 & -20 & 84 \\ 0 & 64 & 0 \end{pmatrix},$$

[065] In the above example, the matrices are combined then multiplied by 64 to convert their coefficients into fixed point binary numbers with 6 bits below the binary point. Other powers of two will work, depending on the precision required and the hardware available. Using a value of 64 in this case results in coefficients that will fit in 8bit bytes with a sign bit. This results in low-cost implimentations where only 8bit arithmetic can be done. In implimentations with 16bit arithmetic a larger multiplier than 64 could be used.

[066] These matrices involve multiplying by 0, by 64 (which is multiplying by one after the fixed point binary shift), by 84 and by 20. Multiplying by 20 can be done with two shifts and an add, multiplying by 84 can be done by three shifts and two adds. Two subtracts are always required after the multiplies. This is simple enough to impliment in hardware or software so it is not necessary to try and find more convenient numbers.

[067] The conversion from sRGB to RGBW can be done in hardware fairly inexpensively. Sub-pixel rendering may require line buffers and filters running at display refresh rates. If a system has hardware SPR, the addition of logic to do RGBW is not appreciably more difficult. In the hardware model, all the RGB values are fetched once for every frame time, converted to RGBW, shifted through line buffers, area resample filtered, sent to the TCON and/or display and forgotten. This system is depicted in Figure 14.

[068] However, in one embodiment of a low cost implementation, SPR may be done in software, as opposed to hardware. Thus, it is reasonable to add RGBW calculations in software as well. In one embodiment, there may be some frame buffers to access. For example, if there is a RGB frame buffer in system memory that application programs write to, then a software driver may convert this data to the sub-pixel rendered version and store it in a hardware frame buffer. Such a sytem is depicted in Figure 15. Optionally, the system could have the driver convert small

rectangular areas that have changed, and not require that the entire display be reconverted every time any change is made.

[069] Often, the software driver may not completely simulate the hardware. For example the software may not have line buffers but does random-access reads to the RGB frame buffer instead. This might require recalculating RGBW values from the RGB values every time they are fetched. For example, in one embodiment, the SPR filters could be 2x3 coefficients. Thus, in this case, each RGB value might be fetched and converted 6 times in the course of re-rendering the area around it.

[070] In one embodiment, determining the chromaticity triangle number could be reduced to 4 compares. Matrix multiply can be done with 5 shifts, three adds and two subtracts. Gamut clamping may require two compares and three divides. Gamut clamping may be done on a small subset of colors and a simple set of 3 tests determines if this step can be skipped. If the processor is fast enough and can do the divisions (or at least, inverse table lookup and multiply) then this may suffice.

[071] However, on a slower processor, with sufficient memory to store another copy of the frame buffer, the time spent converting to RGBW may be reduced by converting every RGB pixel to RGBW only once and storing them in an intermediate frame buffer. For one example, consider a 120x160 by 24bit RGB display. Storing a copy of the RGB frame buffer may take only 58Kbytes. The RGBW intermediate frame buffer would be 77Kbytes. After SPR the hardware frame buffer would only be 39Kbytes. Such a system is depicted in Figure 16.

[072] One additional embodiment might replace the RGBW frame buffer with smaller line buffers. With more software processing, it is possible to build line-buffers of RGBW values similar to the line buffers in typical SPR hardware implementations. Two line buffers the width of the

display might suffice. In this version, the RGB values are only fetched and converted once, then read multiple times out of the line buffers.

[073] While the invention has been described with reference to an exemplary embodiment, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted for elements thereof without departing from the scope of the invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings without departing from the essential scope thereof. Therefore, it is intended that the invention not be limited to the particular embodiment disclosed as the best mode contemplated for carrying out this invention, but that the invention will include all embodiments falling within the scope of the appended claims.